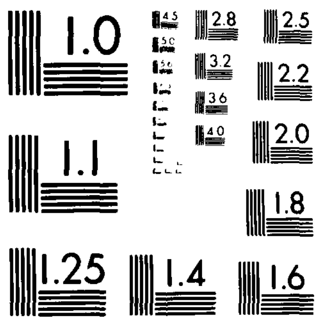
AD-A098 965 YALE UNIV NEW HAVEN CT DEPT OF COMPUTER SCIENCE F/G 9/2
BORIS -- AN EXPERIMENT IN IN-DEPTH UNDERSTANDING OF NARRATIVES.(U)
JAN 81 W LEHNERT, M G DYER, P N JOHNSON N00014-75-C-1111
UNCLASSIFIED RR-188 NL

1 OF 1
AD A
098965

END
DATE
FILMED
6-81
DTIC


1.0
4.5
5.0
5.6
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

LEVEL II

(12)

AD A098965

BORIS -- An Experiment in In-Depth Understanding of Narratives

by

Wendy Lehnert, Michael G. Dyer, Peter N. Johnson,
CJ Yang, and Steve Harley

Research Report # 188

January 1981

Contract N00014-75-C-1111

YALE UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE

81 4 30 020

BORIS -- An Experiment in In-Depth Understanding of Narratives

by

Wendy Lehnert, Michael G. Dyer, Peter N. Johnson,
CJ Yang, and Steve Harley

Research Report # 188

January 1981

This work was supported in part by the Advanced Research Projects Agency under contract N0014-75-C-1111 and in part by the National Science Foundation under contract IST7918463.

(14) RR-188

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered) (9) Research rept.

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER # 188.	2. GOVT ACCESSION NO. AD-A098965	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) (6) BORIS -- An Experiment in In-Depth Understanding of Narratives.		5. TYPE OF REPORT & PERIOD COVERED Technical Report
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) (10) Wendy/Lehnert, Michael G./Dyer, Peter N./Johnson, CJ/Yang [redacted] Steve/Harley		8. CONTRACT OR GRANT NUMBER(s) (15) N00014-75-C-1111, NSF-IST7918463
9. PERFORMING ORGANIZATION NAME AND ADDRESS Yale University - Computer Science Department 10 Hillhouse Avenue New Haven, Connecticut 06520		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Advanced Research Projects Agency 1400 Wilson Boulevard Arlington, Virginia 22209		12. REPORT DATE (11) Jan ~~uary 19~~ 81
		13. NUMBER OF PAGES 74
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Office of Naval Research Information Systems Program Arlington, Virginia 22217 (12) 85		15. SECURITY CLASS. (of this report) unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report)		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Artificial Intelligence
Natural Language Processing
Knowledge Representation
Story Understanding
Question Answering

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

BORIS is a story understanding and question answering system which involves the specification and interaction of many sources of knowledge. Unlike skimmers, which simply extract the "gist" of a story in a top-down manner and ignore everything else, BORIS attempts to understand everything that it reads to as great a depth as possible. This report focuses on how the BORIS program handles a complex story involving a divorce.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

407051

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

-- OFFICIAL DISTIRUBTION LIST --

Defense Documentation Center Cameron Station Alexandria, Virginia 22314	12 copies
Office of Naval Research Information Systems Program Code 437 Arlington, Virginia 22217	2 copies
Dr. Judith Daly Advanced Research Projects Agency Cybernetics Technology Office 1400 Wilson Boulevard Arlington, Virginia 22209	3 copies
Office of Naval Research Branch Office - Boston 495 Summer Street Boston, Massachusetts 02210	1 copy
Office of Naval Research Branch Office - Chicago 536 South Clark Street Chicago, Illinois 60615	1 copy
Office of Naval Research Branch Office - Pasadena 1030 East Green Street Pasadena, California 91106	1 copy
Mr. Steven Wong New York Area Office 715 Broadway - 5th Floor New York, New York 10003	1 copy
Naval Research Laboratory Technical Information Division Code 2627 Washington, D.C. 20375	6 copies
Dr. A.L. Slafkosky Commandant of the Marine Corps Code RD-1 Washington, D.C. 20380	1 copy
Office of Naval Research Code 455 Arlington, Virginia 22217	1 copy

Office of Naval Research Code 458 Arlington, Virginia 22217	1 copy
Naval Electronics Laboratory Center Advanced Software Technology Division Code 5200 San Diego, California 92152	1 copy
Mr. E.H. Gleissner Naval Ship Research and Development Computation and Mathematics Department Bethesda, Maryland 20084	1 copy
Captain Grace M. Hopper NAICOM/MIS Planning Board Office of the Chief of Naval Operations Washington, D.C. 20350	1 copy
Dr. Robert Engelmore Advanced Research Project Agency Information Processing Techniques 1400 Wilson Boulevard Arlington, Virginia 22209	2 copies
Professor Omar Wing Columbia University in the City of New York Department of Electrical Engineering and Computer Science New York, New York 10027	1 copy
Office of Naval Research Assistant Chief for Technology Code 200 Arlington, Virginia 22217	1 copy
Major J.P. Pennell Headquarters, Marine Corp. (Attn: Code CCA-40) Washington, D.C. 20380	1 copy
Computer Systems Management, Inc. 1300 Wilson Boulevard, Suite 102 Arlington, Virginia 22209	5 copies
Ms. Robin Dillard Naval Ocean Systems Center C2 Information Processing Branch (Code 8242) 271 Catalina Boulevard San Diego, California 92152	1 copy

Dr. William Woods 1 copy
BBN
50 Moulton Street
Cambridge, MA 02138

Professor Van Dam 1 copy
Dept. of Computer Science
Brown University
Providence, RI 02912

Professor Eugene Charniak 1 copy
Dept. of Computer Science
Brown University
Providence, RI 02912

Professor Robert Wilensky 1 copy
Univ. of California
Elec. Engr. and Computer Science
Berkeley, CA

Professor Allen Newell 1 copy
Dept. of Computer Science
Carnegie-Mellon University
Schenley Park
Pittsburgh, PA

Professor David Waltz 1 copy
Univ. of Ill at Urbana-Champaign
Coordinated Science Lab
Urbana, Ill 61801

Patrick Winston 1 copy
MIT
545 Technology Square
Cambridge, MA 02139

Marvin Minsky 1 copy
MIT
AI Lab
Cambridge, MA 02139

Professor Negroponte 1 copy
MIT
545 Technology Square
Cambridge, MA 02139

Professor Jerome Feldman 1 copy
Univ. of Rochester
Dept. of Computer Science
Rochester, NY 14627

Professor Nils Nilsson 1 copy
Stanford Research Institute
Menlo Park, CA 94025

Dr. Alan Meyrowitz 1 copy
ONR
Code 437
800 N. Quincy St.
Arlington, VA 22217

Dr. Edward Shortliffe 1 copy
Stanford University
MYCIN Project TC-117
Stanford Univ. Medical Center
Stanford, CA 94305

Dr. Douglas Lenat 1 copy
Stanford University
Computer Science Department
Stanford, CA 94305

Dr. M.C. Harrison 1 copy
Courant Institute Mathematical Science
New York University
New York, NY 10012

Dr. Morgan 1 copy
Univ. of Penn
Dept. of Computer Science & Infor Science
Philadelphia, PA 19104

BORIS -- An Experiment in In-Depth Understanding of Narratives

by

Wendy Lehnert, Michael G. Dyer, Peter N. Johnson,
CJ Yang, and Steve Harley

Abstract

BORIS is a story understanding and question answering system which involves the specification and interaction of many sources of knowledge. Unlike skimmers, which simply extract the "gist" of a story in a top-down manner and ignore everything else, BORIS attempts to understand everything that it reads to as great a depth as possible. This report focuses on how the BORIS program handles a complex story involving a divorce.

Special acknowledgements to Mark Burstein and Tom Wolf for their valuable contributions to the theory, design, and implementation of the BORIS system. Thanks also to Judy Martel who edited the manuscript, and to Margot Flowers, who made valuable suggestions on an earlier draft.

Table of Contents

List of Figures

1. INTRODUCTION

1.1 Background

Previous language understanding systems at Yale have been designed to process a great many stories. For example, FRUMP [2] was able to read about 10% of stories off the UPI news wire, and IPP [6] has read over one hundred newspaper stories concerning terrorism. In order to accomplish such robustness, however, these programs have paid a price in the depth of their understanding. That is, they "skim" at various levels, seeking to extract the "gist" of a story. For instance, given an earthquake "sketchy script", FRUMP could extract from a story the time, place, magnitude and number of casualties caused by the quake. Everything else would be ignored. For instance, consider the hypothetical "human interest" segment below:

> Last night Gov. Jerry Brown was awoken by a violent earthquake. He was thrown from his bed and landed on his dog, who then bit him. Authorities described the earthquake as registering ...

Similarly, IPP achieves its robustness by only understanding those portions of a story that relate directly to its knowledge of terrorism. IPP knows a lot about terrorism, but little about other things.

In contrast, BORIS represents an effort to see what is involved in understanding a few stories in great depth. The first story used by the BORIS project is described in [3] and involves a kidnapping which disrupts a farewell party, but then turns out to be a hoax.

This report describes the results of building a program to process a more complex story, involving a divorce case.

1.2 I/O Behavior

What follows is an actual understanding and question answering session with the BORIS program running on the divorce story: (Input to the program is in lower case; output in uppercase.)

```
 TOPS-20 Command Processor 4(560)-1
@BORIS-DEMO.EXE.5

Top-Level of LISP
*(EXPFS 50000)
*(BORIS ´DIVORCE-STORY)

PROCESSING PARAGRAPH 1:

    Richard hadn´t heard from his college
    roommate Paul for years.
    Richard had borrowed money from Paul
    which was never paid back.
    but now he had no idea where to find his old friend.
    When a letter finally arrived from San Francisco,
    Richard was anxious to find out how Paul was.

READY TO ACCEPT Q/A :

    > What happened to Richard at home? <
    RICHARD GOT A LETTER FROM PAUL.

    > Who is Paul? <
    RICHARD ´s FRIEND.

    > Did Richard want to see Paul? <
    YES, RICHARD WANTED TO KNOW HOW PAUL WAS.

    > Had Paul helped Richard? <
    YES, PAUL LENT MONEY TO RICHARD.

    > Why didn´t Richard pay Paul back? <
    RICHARD DID NOT KNOW WHERE PAUL WAS.

    > How did Richard feel when the letter appeared? <
    RICHARD FELT GLAD BECAUSE PAUL AND HE WERE FRIENDS.

    > go <
```

```
PROCESSING PARAGRAPH 2:

    Unfortunately, the news was not good.
    Paul's wife Sarah wanted a divorce.
    She also wanted the car, the house,
    the children, and alimony.
    Paul wanted the divorce, but he didn't want to
    see Sarah walk off with everything he had.
    His salary from the state school
    system was very small.
    Not knowing who to turn to,  he was hoping
    for a favor from the only lawyer he knew.
    Paul gave his home phone number in case
    Richard felt he could help.

READY TO ACCEPT Q/A :

    > What was the letter about? <
    PAUL AND SARAH WERE GETTING A DIVORCE.

    > Did Paul write Sarah? <
    NO, IT WASN'T SARAH, IT WAS RICHARD WHO GOT THE LETTER.

    > Why was Paul upset about the divorce? <
    PAUL AND SARAH WERE FIGHTING OVER THE FAMILY POSSESSIONS.

    > What did Paul do for a living? <
    PAUL WAS A TEACHER.

    > What did Sarah want? <
    SARAH WANTED TO HAVE THE CAR AND THE HOUSE AND THE KIDS AND
    THE ALIMONY.

    > Why did Paul write to Richard? <
    PAUL WANTED RICHARD TO BE HIS LAWYER.

    > go <

PROCESSING PARAGRAPH 3:

    Richard eagerly picked up the phone and dialed.
    After a brief conversation,
    Paul agreed to have lunch with him the next day.
    He sounded extremely relieved and grateful.

READY TO ACCEPT Q/A :

    > Why did Richard call Paul? <
    RICHARD WANTED TO ARRANGE A MEETING WITH PAUL.

    > Did Richard contact Paul? <
    YES.  RICHARD CALLED PAUL.

    > How did Richard contact Paul? <
    BY PHONE.

    > How did Paul feel when Richard called? <
    PAUL WAS HAPPY BECAUSE RICHARD AGREED TO BE PAUL'S LAWYER.

    > go <

PROCESSING PARAGRAPH 4:

    The next day, as Richard was driving to the restaurant,
    he barely avoided hitting an old man on the street.
    He felt extremely upset by the incident,
    and had three drinks at the restaurant.
    When Paul arrived Richard was fairly drunk.
    After the food came, Richard spilled a cup of coffee on Paul.
    Paul seemed very annoyed by this so Richard
    offered to drive him home for a change of clothes.

 READY TO ACCEPT Q/A :

    > Why did Richard eat out with Paul? <
    RICHARD WANTED TO DISCUSS THE DIVORCE CASE WITH PAUL.

    > What happened to Richard on the way to the restaurant? <
    RICHARD ALMOST RAN OVER AN OLD MAN.

    > Why did Richard get drunk? <
    RICHARD WAS UPSET ABOUT ALMOST RUNNING OVER THE OLD MAN.

    > What happened at the restaurant? <
    RICHARD SPILLED COFFEE ON PAUL.

    > How did Richard feel when the coffee spilled? <
    RICHARD WAS UNHAPPY.

    > Why did Richard spill the coffee? <
    RICHARD WAS DRUNK.

    > When did Richard almost hit the old man? <
    WHILE RICHARD WAS DRIVING TO THE RESTAURANT.

    > Where did Richard have lunch with Paul? <
    AT A RESTAURANT.

    > go <

PROCESSING PARAGRAPH 5:

    When Paul walked into the bedroom and found
    Sarah with another man he nearly had a heart attack.
    Then he realized what a blessing it was.
    With Richard there as a witness,
    Sarah's divorce case was shot.
    Richard congratulated Paul and
    suggested that they celebrate at dinner.
    Paul was eager to comply.

READY TO ACCEPT Q/A :

    > Why did Richard drive Paul home? <
    SO PAUL COULD CHANGE HIS CLOTHES.

    > What happened to Paul at home? <
    PAUL CAUGHT SARAH COMMITTING ADULTERY.

    > How did Paul feel? <
    PAUL WAS SURPRISED.

    > Why did Sarah lose her divorce case? <
    SARAH COMMITTED ADULTERY.

    > Why did Richard congratulate Paul? <
    PAUL WON HIS DIVORCE CASE.

    > go <


TEXT PROCESSING COMPLETED ON:  DIVORCE-STORY
```

1.3 Problems, Problems, Problems

To give some idea of what BORIS is up against in trying to understand this story, we are going to take a first pass through the story and point out just a few of the problems involved:

(a) Richard hadn't heard from his college roommate Paul in years.

Here BORIS is told a non-event. What should BORIS do with this? Simply storing an event as not having occurred is inadequate. When people read: "John walked into the room and Mary wasn't there." they infer that John wanted to see Mary and had been expecting her there. They do not simply store in memory the fact: FALSE [LOC [MARY,OFFICE]].

(b) ... which was never paid back.

This phrase also refers to a non-event, but what is its effect upon processing? Many people who read the story said that Richard was willing to help Paul because Richard felt obligated to Paul. Therefore, this non-event must be understood in terms of the relationship between Paul and Richard. In addition, it should be noted that this phrase never explicitly mentions WHO failed to pay WHAT to WHOM, yet people never notice this, and automatically fill these roles in from their knowledge about the relationship between borrowings and repayments.

(c) ... but now he had no idea where to find his old friend.

How do we go about parsing an expression like (c)? And once parsed, how do we represent its semantics? (I.e. what representation should the parser have produced?) The word "had" does not refer to physical possession. The word "old" does not refer to the age of Richard's friend. How do we capture the meaning of "finding" someone? Also, what is the connection of (c) to (b) before it?

(d) When a letter finally arrived from San Francisco ...

People assume that the letter is from Paul even though this is never explicitly stated.

(e) Unfortunately, the news was not good.

People assume that the entire second paragraph refers to information contained in the letter and that Richard is reading this letter. But we are never explicitly told when the letter context is entered (or exited).

(f) Paul's wife Sarah wanted a divorce.

How is "divorce" represented in memory? It must, at the very least, refer both to marriage and to legal disputes. Otherwise, the mention of Paul's need for a lawyer would make no sense.

(g) ... but he didn't want to see Sarah walk off with everything he had.

How is this to be parsed? To represent "walk off" as physical movement is inadequate. "See" doesn't refer here to vision. How do we represent "everything" or find its referent?

(h) His salary from the state school system was very small.

What does (h) have to do with the story? People immediately realize the connection between a small salary and lawyers' fees, alimony, etc. But how is this connection to be formalized in a computer program?

(i) Not knowing who to turn to, he was hoping for a favor from the only lawyer he knew.

How are "who to turn to", "hoping", "favor", "only" to be parsed and represented? What must BORIS know about lawyers in order to understand why "lawyer" has been mentioned? Notice, also, that the story never explicitly states that Richard is the lawyer being referred to.

(j) Richard picked up the phone and dialed.

The story never explicitly states who Richard dialed (or engaged in conversation).

(k) He sounded extremely relieved and grateful.

What is to be done with affects such as "relieved" and "grateful"? Furthermore, which character is the "he" that feels this way?

(l) ... he barely avoided hitting an old man on the street.

How does "barely avoided" affect processing? How does BORIS realize that "hitting" here refers, not to a fist fight, but rather to a vehicle accident which almost occurred?

(m) When Paul arrived Richard was fairly drunk.

The story never explicitly states where it is that Paul has arrived. People, however, automatically interpret Paul's arrival in the context of their arrangement to meet. But how?

(n) ... so Richard offered to drive him home for a change of clothes.

Why did Richard make this offer? Why does Paul have to change clothes? Connections must be made between these events in the story if such questions are to be answerable during Q/A .

(o) When Paul walked into the bedroom ...

What is Paul doing in the bedroom? How did Paul get there? Notice that scene changes are made implicitly in narratives. Nor are we explicitly told that Richard drove Paul home, only that he offered to.

(p) ... and found Sarah with another man he nearly had a heart attack.

The story never explicitly states that Sarah was having an affair. This must be inferred. Also, in this case, "heart attack" refers to shock or surprise, not to a cardiac arrest.

(q) Then he realized what a blessing it was.

What is "blessing"? What does the word "it" refer to? What is the effect of this sentence upon processing? What is constructed in memory?

(r) With Richard there as a witness, Sarah's divorce case was shot.

What does BORIS need to know about "witness" to successfully understand this sentence? The word "case" here is not a container or unit of measurement, as in "case of beer". Also, "was shot" is metaphorical.

(s) Richard congratulated Paul and suggested they celebrate at dinner.

How does BORIS represent "congratulated"? Why did Richard make this suggestion? The story never says that Paul would win, simply that Sarah would lose. But how is "losing a legal case" to be captured in a computer program?

Notice that, unlike previous systems such as SAM [1] and PAM [16], BORIS must deal with the specification, application and interaction of many sources of knowledge. Among others, the divorce story contains:

object primitives	(phone, letter, coffee, clothes)
scripts	(drive, dine)
settings	(bedroom, restaurant)
goals	(wanting a divorce)
plans	(getting a lawyer)
affects	(eager, anxious, grateful)
roles	(lawyer, teacher)
themes	(friendship, marriage)
physical states	(drunk)
events	(spilling the coffee)
social acts	(petition judge in court)

In addition to these, which have been described elsewhere [13], [14], [7], BORIS employs other types of knowledge. These knowledge structures are discussed in section 2.

1.4 Overall Organization and Control

The BORIS story understanding system is comprised of four basic processing units: the conceptual analyzer (the parser), the event assimilator (EA), the question answering (Q/A) module, and the English generator. These four modules interact to form a complete story understanding system. All input to (and output from) the program is in English.

Processing begins with the parser program which reads the English text from left to right. This module generates conceptual dependency (CD) structures [15] which capture the semantic content of the phrases and sentences that it is reading. These CD structures are kept in a short-term structure called the Working Memory buffer (WMB).

The EA examines the concepts in WMB as they are produced.

It is this module's responsibility to produce an episodic memory representation of the story. Each WMB concept must be understood in the context of everything that has been read so far. In order to do this, the event assimilator must consult a wide body of world knowledge as well the representation that it has produced for the preceding portion of the story.

The Q/A module in BORIS invokes the same conceptual analyzer to read the English questions. As concepts are produced to represent the question, the episodic story representation as well as the associated semantic memory knowledge structures are searched to find the answer. The question answerer uses search heuristics and actually modifies episodic memory during the search process. Inferences are thus being made at question answering time. When the answer to the question is found, it is formulated into the appropriate canonical form and passed on to the English generator.

The generator takes both CD structures and episodic memory structures as input, and produces English sentences as output.

The understanding process employed by BORIS is actually much more integrated than this description implies. The parser must frequently consult both episodic and semantic memory as well as its lexicon to understand the phrases and sentences of the story as they are read. In order to do this, it must elicit the help of the EA to index the memory structures that are relevant. As a result, the parser does in fact sometimes produce direct references to knowledge structures and episodic memory units in its representation. Thus there is a great deal of communication and interaction between the parser and the event assimilator.

The question answering program and the English generator are also involved with the BORIS memory organization. As shown in Figure 1-1, all four of the BORIS modules must access and search both episodic and semantic memory. Consequently, the entire BORIS project shares a large body of code.

```
+---------------------+
| CONCEPTUAL ANALYSIS | <------------------ +
|        (C/A)        |* * *                |
+---------------------+    *                |
     / \ |      ...........*....            |
      |  |      :--------------:            |
      |  |      :   SEMANTIC   :            |
      |  |      :    MEMORY    :            |
      |  |      :-----    -----:   +--------------------+
      |  |      :              :* * *| QUESTION ANSWERING |
      |  |      :   EPISODIC   :   |       (Q/A )       |
      |  |      :    MEMORY    :   +--------------------+
      |  |      :--------------:      / \          |
      |  |   * * *.............:       |           |
      | \ /  *                *        |          \ /
+-----------*--------+        *        |   +------------------+
| EVENT ASSIMILATION |        * * * * * * * *| GENERATOR        |
|       (EA)         |-----------------+   |   (E/G)          |
+--------------------+                     +------------------+
```

Figure 1-1: OVERALL ORGANIZATION: Dashed lines represent control flow while starred lines represent data flow.

2. KNOWLEDGE REPRESENTATION AND EPISODIC MEMORY

The most important knowledge structures in BORIS are MOPs, META-MOPs [10] and TAUs. A MOP (Memory Organization Packet) is a configuration of CDs formed into a discrete knowledge structure by a standard set of semantic links (which are described at length in [3]). TAUs (Thematic Affect Units) will be discussed in section 2.3.

2.1 MOPs and META-MOPs

Unlike scripts, which only consider the actions performed by characters within a given setting, MOPs also include their goals and intentions. Associated with each MOP are a set of MOP-links, which dynamically specify the relationship of one MOP to another during story understanding, and expectations concerning what may happen next. In addition, MOPs serve as indices into episodic memory. Episodes in BORIS are reconstructed dynamically out of several MOPs, META-MOPs and TAUs. This will discussed later.

Figure 2-1 shows an example of the MOP which represents BORIS's knowledge about borrowing and lending objects. All MOP diagrams have been simplified for the sake of clarity. For instance, each node in the MOP points to a Conceptual Dependency (CD) construction. However, each CD has been replaced by a mnemonic name which indicates the event or goal which is involved. The links i, m, and a represent intention, motivation and achievement, respectively. These and others not shown here are described at length in [3]. By convention, events and plans are represented in the central column while goals are specified under each role in the outer columns of the diagram:

```
-------------------------------------------------------
|                                                     |
| BORROWER                             LENDER         |
| --------                             ------         |
|                 i                                   |
| WANT-OBJECT ------                                  |
|          |      |              m                    |
|          |    ASK-FOR-OBJECT --------               |
|          |                          |               |
|          |  a               i       |               |
|          |-------      ---------CONVINCED-TO-LEND   |
|                 |       |                           |
|             m   |       |    m                      |
|          ------GIVE-OBJECT-----------               |
|          |                          |               |
|          |      i                   |               |
| WANT-TO-RETURN-----                  WANT-RETURNED  |
|                   |                     |           |
|                   |                 a   |           |
|                GIVE-OBJECT-BACK ---------           |
|                                                     |
|                                                     |
| MOP-links:   /---- IPT-FRIENDSHIP ----> MM-FAVOR    |
|             *                                       |
|              \----  ...  --> MM-BUSINESS-CONTRACT   |
-------------------------------------------------------
```

Figure 2-1: M-BORROW

M-BORROW captures the essentials of lending. The borrower wants something, so he asks the lender for it. If the lender gives it, then the lender will want it back later and the borrower will feel obligated to return it. BORIS uses this knowledge structure to understand the borrowing event in the first paragraph of the divorce story. When a lending occurs, BORIS uses M-BORROW to create an expectation that the object borrowed will be returned at some point in the future. When BORIS reads "which was never paid back", it can understand "paid back" in terms of M-BORROW and automatically infer the missing roles.

In addition, M-BORROW contains a number of MOP-links, which examine the context to determine which META-MOPs may be relevant. If the situation is informal (for example, the borrower and lender are friends) then MM-FAVOR is activated. (The prefix "IPT" refers to interpersonal themes.)

BORIS knows that friends like to: a) socialize together, b) know how the other is doing, and c) do things for one another. If these conditions are violated too often, then the friendship may be terminated.

MOP-links are uni-directional. In other words, M-BORROW may activate MM-FAVOR but the reverse is not true. This is because almost any activity could count as a favor. For example, taking someone to the movies might constitute a favor. MM-FAVOR cannot be expected to know of all such possibilities. MM-FAVOR is shown in Figure 2-2.

It is important that BORIS understand the lending event at both the MOP and META-MOP levels. First, each knowledge structure captures a different set of expectations. At the M-BORROW level, BORIS expects Richard to repay the money to Paul. His expectation is rather specific and, incidentally, is never fulfilled in the story. At the MM-FAVOR level, however, the expectation that Richard will help Paul is later fulfilled when Richard agrees to represent Paul in court. This expectation for

```
-------------------------------------------------------
|                                                     |
| FRIEND-A                            FRIEND-B        |
| --------                            --------        |
|                 i                                   |
| WANT-FAVOR -------                     THEMATIC     |
|          |        |               m    OBLIGATION   |
|          |        ASK-FOR-FAVOR -------    |        |
|          |        (invoke theme)      |    | m      |
|          |                            |    |        |
|          | a                 i        |    |        |
|          |---------      ----------- PERSUADED      |
|                   |      |                          |
|              m    |      |                          |
| THEMATIC ------- DO-FAVOR (agency)                  |
| OBLIGATION                                          |
-------------------------------------------------------
```

Figure 2-2: MM-FAVOR

a return-favor would exist whether Richard had repaid Paul or not. Thus, MM-FAVOR represents "vague" expectations, in which we assume Richard will do something for Paul, but we're not sure exactly what.

META-MOPs are also important because they represent knowledge about the event which can not be captured at the MOP level. For example, what if the story had read:

> Richard had borrowed $800 from Mr. Jones
> at the bank, which he had never paid back.

This lending is definitely NOT a favor, but rather a business contract, and the inferences following from Richard's failure to pay are very different.

Some MOPs are difficult to specify. In BORIS, divorce is a MOP. But how should a divorce be represented? What goes into M-DIVORCE? Each time we tried to specify the content of M-DIVORCE, we ended up talking about other knowledge structures, such as marriage, adultery, custody battles, etc. In fact, very little is in M-DIVORCE, which consists mainly of MOP-links to other knowledge structures. Figure 2-3 shows a portion of what BORIS knows about divorces.

```
-----------------------------------------------------
|                                                   |
| SPOUSE-A                          SPOUSE-B        |
| --------                          --------        |
|                   possible                        |
| WANT-TERMINATE -------------------- WANT-TERMINATE |
| MARRIAGE          conflict        MARRIAGEATION   |
|                      |                            |
|                  CONTESTED                        |
|                   DIVORCE                         |
|                                                   |
|                   possible                        |
| WANT-FAMILY  ---------------------- WANT-FAMILY   |
| POSSESSIONS       conflict        POSSESSIONS     |
|                      |                            |
|                   CONTESTED                       |
|                   SETTLEMENT                      |
|                                                   |
|   MOP-link:  * ---- conflict ----> MM-LEGAL-DISPUTE |
|                                                   |
-----------------------------------------------------
```

Figure 2-3: M-DIVORCE

When many people hear about a divorce, they do not necessarily think about lawyers and legal disputes. If people thought about all the knowledge structures potentially related to a given event, they would be overwhelmed. Yet, if a lawyer is mentioned in the context of a divorce, people immediately understand why this is relevant. BORIS accomplishes this by means of its MOP-links. If BORIS reads that the spouses have a major goal conflict, then it activates MM-LEGAL-DISPUTE. This knowledge structure represents what BORIS knows about disputes which are resolved through the process of petitioning a judge in court [14]. Since a lawyer is needed to perform these petitions, its relevancy can then be understood. MM-LEGAL-DISPUTE is described in Figure 2-4.

```
-----------------------------------------------------
|                                                   |
| DISPUTEE-A                           DISPUTEE-B   |
| ----------                           ----------   |
|                    conflict                       |
| GOAL-A ----------------------------- GOAL-B       |
|  | |                 |                  | |       |
|  | |  i           DISPUTE          i    | |       |
|  | |---------|                |-------| |       |
|  |           |                |           |       |
|  |     GET-LAWYER-A    GET-LAWYER-B       |       |
|  |     (petition)      (petition)         |       |
|  |                                        |       |
|  |                  JUDGE                 |       |
|  |      a           ------                |       |
|  |---------------- DECISION-A      a      |       |
|                    DECISION-B -----------         |
|                                                   |
-----------------------------------------------------
```

Figure 2-4: MM-LEGAL-DISPUTE

That is, two individuals have goals in conflict. Each lawyer presents a case to the judge. The judge decides favorably in terms of one and unfavorably in terms of the other. The merits of the case depend upon the nature of the dispute. For example, in contractual disputes, the judge is likely to decide unfavorably for the disputee who broke the contract. (This is how BORIS realizes that Sarah's adultery has caused her to lose her case.)

2.2 Memory Overlays

When a MOP or META-MOP is activated via a MOP-link, the relationship between the activating knowledge source and the activated knowledge source must be known. For example, BORIS must realize not only that the divorce has caused a legal dispute, but also that the issue under dispute within MM-LEGAL-DISPUTE involves a property and custody settlement. BORIS accomplishes this by "overlaying" one knowledge structure on another. An "overlay" occurs by specifying which component in one knowledge structure is equivalent to another component within some other knowledge structure. Figure 2-5 shows how MM-LEGAL-DISPUTE, M-LAWYER, MM-PROF-SERVICE, M-DIVORCE and MM-FAVOR are overlaid.

```
   MM-LEGAL-DISPUTE       MM-PROF-SERVICE    MM-FAVOR
 --------------------    -----------------  ------------
|                   |   |               |  |           |
| * * DISPUTE       |   | DO-SERVICE * * |  |    ...    |
| *      ...        |   |     ...      * |  |           |
| *                 |   |              * |  | DO-RETURN |
| * GET-LAWYER-A *  |   |  PAY-MONEY   * |  | FAVOR  * *|
| *            *    |   |              * |  |          *|
--*------------*--       -------------*--   ----------*-
  *            *                      *               *
  *            * * * *                * * * * * * * * *
  *                  *                              *
  *      M-DIVORCE   *         M-LAWYER             *
  *   -------------- *     ----------------         *
  *   |     ...     | * * * * * TAKE-CASE  |        *
  *   |             |      |     ...       |        *
  * * * * CONTESTED |      |               |        *
      |  SETTLEMENT |      |   PETITION * * * *
      ---------------      ----------------
```

Figure 2-5: MEMORY OVERLAYS

The CONTESTED SETTLEMENT in M-DIVORCE is the DISPUTE at issue in MM-LEGAL-DISPUTE. The need-for-a-lawyer aspect of the dispute (i.e. GET-LAWYER-A) points to what BORIS knows about lawyers. As Richard prepares Paul's case (i.e. PETITION), he is both performing a professional service and doing an old friend a favor.

This overlay scheme has several advantages:

1. Each knowledge structure can be specified independently of other knowledge structures, and then later on become connected to other structures via MOP-links.

2. Each knowledge structure need know only that which is is directly relevant to it. For example, M-LAWYER contains role information about what lawyers do (e.g. meet with the client, gather evidence, etc.). However, the fact that the lawyer is performing this task as a means of earning his living, is captured by MM-PROF-SERVICE, which represents the contractual exchange of money for ANY type of professional service. Thus, MM-PROF-SERVICE need not be copied for doctors, plumbers, etc. This allows for economy of storage [10].

3. Each knowledge source need not be activated unless an event occurs which is directly relevant to it. For example, people do not normally think of the need to pay the lawyer unless payment (or failure or refusal to pay) is explicitly mentioned.

4. A given event can be understood from several perspectives. When Richard and Paul meet at lunch, we have both two old college buddies getting together to socialize after not having seen each other in a long time, and a lawyer and a client meeting to discuss a

legal case. Since people have no troubles maintaining multiple perspectives -- e.g. Paul is a spouse, college friend, teacher, disputee and legal client -- we want BORIS to have the same capability.

2.3 TAUs

Not every type of knowledge can be organized around MOPs or META-MOPs. For instance, what should we do with the following sentence in the story?

Not knowing who to turn to, he was hoping for a favor from the only lawyer he knew.

It is evident that Paul perceives himself to be in serious trouble. He doesn't know what to do. He's not sure that Richard will help him and he doesn't know anyone else able to help. Later we read:

He sounded extremely relieved and grateful.

This comes after Richard agrees to take the case. It is the knowledge we have about Paul's dilemma that allows us to determine that "he" in the sentence above is referring to Paul and not Richard.

BORIS makes use of a knowledge structure called a TAU (Thematic Affect Unit). TAUs are used to capture aspects of both Affect Units [9] and TOPs [10]. In the case above, TAU-DIRE-STRAITS is used by BORIS to aid in understanding Paul's situation. This TAU represents knowledge about how people feel and react when they are in a crisis. Thus, the "extreme relief" felt by Paul can be predicted using TAU-DIRE-STRAITS.

Below are some of the TAUs used in the divorce story and the events which they help represent:

TAU-DIRE-STRAITS - Paul doesn't know who to turn to.

TAU-CLOSE-CALL - Richard nearly kills an old man.

TAU-REG-MISTAKE - Richard spills coffee on Paul.

TAU-RED-HANDED - Paul catches Sarah in bed.

TAU-BROKEN-OBLIGATION - Sarah has been having an affair.

TAU-HIDDEN-BLESSING - Paul realizes his problems are solved.

TAUs have the following components which help distinguish them from MOPs:

1. They are thematic in nature. That is, they often capture knowledge which people represent in adages. These adages tell about how people respond to failures in planning. For example, "A friend in need is a friend indeed" sums up some aspects of TAU-DIRE-STRAITS, while "Every cloud has a silver lining" sums up some aspects of TAU-HIDDEN-BLESSING.

2. TAUs have a strong affective component. For instance, Paul is very anxious about getting Richard to be his lawyer; Richard feels very upset over his near accident on the road; Paul is shocked to find Sarah with another man, etc.

3. TAUs involve variations from the standard goal/plan processing handled by MOPs. In most TAUs, a plan goes

wrong (or almost goes wrong). That is, most TAUs arise as unexpected events in a narrative. The near accident, the spilled coffee, and the discovered affair, are all surprising to the characters involved.

4. Unlike MOPs, TAUs are activated by idiosyncratic indices which are recognized in a bottom-up manner. For example, TAU-RED-HANDED is activated when a goal to violate a norm, which requires secrecy for its success, fails during plan execution due to a witnessing.

5. TAUs are sensitive with respect to point of view. For example, catching someone red-handed is a very different experience than that of being caught red handed. Thus, the index into memory would depend upon which role is being focused on.

2.4 Semantic vs. Episodic Memory

We consider the dichotomy between semantic and episodic memory to be artificial. This is especially true in systems systems which learn and programs which model long-term human memory [5].

We believe that semantic memory is created through episodic experience and, thus, there can be no clean distinction between the two. However since BORIS is not a learning project, it is convenient to talk about what BORIS knows before it reads the divorce story as its "semantic" memory, and what it has constructed after reading the story as its "episodic" memory.

BORIS´s episodic memory consists of fragmentary instantiations of MOPs, META-MOPs and TAUs. The central organizing concept is that of an "episode", which consists of an instantiation of several interrelated knowledge structures. An episode is analogous to a script except that it is dynamically constructed out of other relevant knowledge structures rather than statically organized. Figure 2-6 shows a simplified "picture" of story memory after having read the divorce story. Each episode is underlined by "====":

By examining this diagram, we can see that: Richard received a letter about a divorce which required getting a lawyer as a favor. A meeting was arranged by phone and an accident almost occurred while on the way to the meeting, etc.

Each episode organizes those knowledge structures which are related to a given event. For example, EP7 represents the event of Paul catching Sarah breaking their marriage contract by having an affair. One way episodes are connected is indirectly through goals. For example, the goal of driving is instrumental to the goal of meeting with Paul. Episodes can also be interconnected via TAUs. TAU-HIDDEN-BLESSING represents those situations in which an event occurs that can be viewed from both positive and negative perspectives. So Sarah´s adultery is bad from the marriage perspective, but good from the legal perspective.

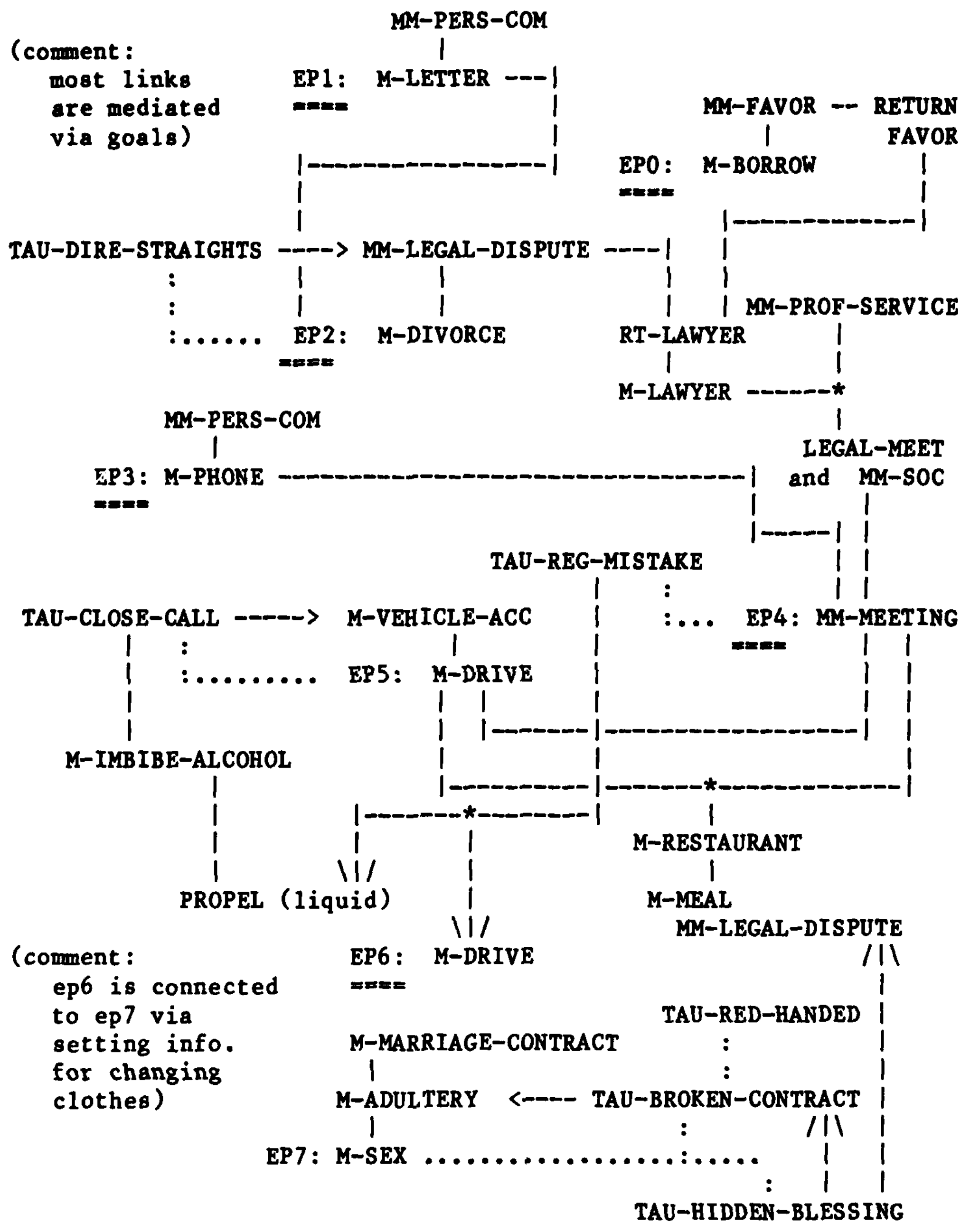


Figure 2-6: EPISODIC MEMORY

3. PARSING

Parsing here refers to the task of conceptual analysis, which involves extracting the conceptual content of each English sentence. BORIS makes use of a demon-based parser designed by Michael Dyer. It is an outgrowth of Reisbeck's request-based parser [12]. BORIS contains a word and phrasal lexicon, where each lexical entry has one or more "demons" associated with it. Demons represent expectations and other processing information. Demons may "spawn" other demons and each demon is in control of its own "life" and "death". (Section 8.2 contains an execution trace of the parser during Q/A.)

3.1 Depth of Parsing

The goal of conceptual analysis is to map natural language input to a representation of its meaning. However, since high-level memory structures (e.g. TAUs and MOPs) as well as low-level ones (e.g. CDs) are used for story representation in BORIS, we have to determine what level of representation the parser should produce.

Choosing the proper level of representation is important in BORIS for the following reason: Since the memory searching process for parsing during Q/A (question answering) time and EA (event assimilation) time are essentially the same, (the differences are discussed in section 5) only one parser is employed in the system. If the parser does not make enough inferences and understands the input at a level lower than it should, then it will have trouble constructing the right index for memory search during Q/A. , Consider:

> When Paul walked into the bedroom and found Sarah with another man, ...

The parser could understand this sentence at two different levels:

1. It could just interpret the above sentence as Paul seeing Sarah in the proximity of another man in a bedroom, and then let EA figure out that Sarah was having an affair and committing adultery.

2. Alternatively, the parser could understand the above sentence directly as Sarah having an affair with another man. I.e. the inferences would be done by the parser instead of by EA.

The final memory node created by EA in the story representation should be the same for both cases.

Which level is more appropriate? The answer depends on the parsing process required during Q/A. Consider the following question asked during a subsequent Q/A session:

> **How did Paul feel when he found Sarah with another man in the bedroom?**

If the parser understands the question at the first level, the memory search will be based on mere physical proximity and the search will not succeed because the event has not been stored that way in the first place. Therefore, Q/A would need another level of processing to make the inference about adultery from "Sarah and another man in the bedroom". On the other hand, if the parser produces a representation at the second level, then neither EA nor Q/A needs to worry about making such inferences. Instead of giving the burden to both EA and Q/A , we leave it to the parser whenever the same kind of inference is required at both EA time and Q/A time. (This is actually a good argument for further integrating both EA and the parser into a single process. See section 7.1.)

3.2 Parser/Memory Interaction

Parsing is a process of knowledge application. A tremendous amount of knowledge is used to correctly understand even a simple sentence. This knowledge must include the episodic memory constructed by EA -- in addition to lexical, linguistic, and general world knowledge.

What does it mean to use episodic memory in parsing? Why is it useful for parsing? The following sections illustrate the usefulness of memory interactions at EA time.

3.2.1 Accessing Static Episodic Structures

3.2.1.1 MOPs

The first memory interaction the parser performs is to examine the instantiated memory nodes created by EA to facilitate inferences. This type of memory lookup is frequently used to infer slot fillers because often a sentence leaves out some information mentioned previously. So the parser has to retrieve information from the memory. Take a simple case from the story:

Richard had borrowed money from Paul ...

At this point, the parser has no difficulty in producing a correct representation will successfully interpret it as a borrowing event which occured some time in the past. However, the second part of the sentence

which was never paid back.

does not explicitly tell us who paid what back to whom. This problem can easily be solved with the help of memory. The semantic memory says the borrower pays back the borrowed object to the lender; the episodic memory says the borrower is Richard, the lender is Paul, and the borrowed object is the money. Thus, when roles are missing, the parser searches the most recent active MOP for an expected event, and if a match occurs, it uses the bindings already available in episodic memory to infer the missing roles.

3.2.1.2 TAUs

Memory is also helpful for resolving pronouns. Here is an example from the story:

a) He sounded extremely relieved and grateful.

What does "he" refer to? There seem to be two solutions to this problem. First, we can look at the structure of the previous sentence:

Paul agreed to have lunch with him the next day.

A simple minded syntactic rule can be formulated to resolve the pronoun:

```
IF the first participant in the current sentence
   is referred to by a pronoun,
  AND there is more than one possible referent
   satisfying gender agreement,
THEN the pronoun refers to the first participant
   mentioned in the last sentence.
```

Using this rule, BORIS will correctly infer that "he" is Paul. Or can we? What if the story had read:

> After a brief conversation, Richard arranged to have a meeting with Paul to discuss the case the next day. He sounded extremely relieved and grateful.

Here, although the first character mentioned in the first sentence is Richard, the "he" in the second sentence still refers to Paul. So a more powerful and general solution is needed. This requires accessing the 'appropriate' knowledge structure -- i.e. one which can determine the referent. But how is this to be done?

Realizing who "he" refers to involves understanding what "relieved" and grateful" imply from a goal-processing point of view. In BORIS, affects are organized around TAUs. Furthermore, "relieved" and "grateful" refer to the following affect/goal situations:

```
If x feels "relieved"
Then x has achieved a goal which
  x felt might not be achieved,
  or x had a momentary preservation goal
  activated by a 'fleeting' threat.

If x feels "grateful"
Then an agent, y, has either
  achieved a goal for x,
  or x has found an agent to
  achieve that goal.
```

But these situations are exactly what is captured by TAU-DIRE-STRAITS, which was instantiated earlier in memory (see section 2.3). In TAU-DIRE-STRAITS it was Paul who was being threatened with financial loss and who feared that Richard may not help. It was Richard who achieved Paul's goal of finding a lawyer to represent him. Thus, by examining this TAU, the parser finds that Paul is the one in trouble, and is the likely referent for "he" above.

3.2.1.3 Scenarios

The scenario map [3], an episodic memory access structure discussed in 4.1.3 gives helpful hints on making important inferences. Recall this example:

> When Paul walked into the bedroom and found Sarah with another man...

First, when the parser reads ´person A with person B´, it tries to find an activity that is associated with A and B. One way is to look at the locations of A and B, as in:

1. Bill saw John with Mary in a theater.
2. Bill saw John with Mary in a restaurant.

Sentence 1. implies that John and Mary were watching movies, while 2. implies John and Mary were eating in a restaurant. Using this same strategy, the parser infers that Sarah was having sex, because the current scenario is BEDROOM and M-SEX is the social activity which occurs in this setting.

Word sense disambiguation is a major parsing problem. The approach of the BORIS parser is to query the memory for clues. Consider the following example from the story:

> The next day, as Richard was driving to the restaurant, he barely avoided hitting an old man on the street.

The second part of this sentence has at least two possible interpretations:

1. Richard nearly had a fight with an old man.

2. Richard's car nearly hit an old man.

There are two ways that the word "hit" can be disambiguated in this example. The parser, upon reading the word "hitting", pauses to look in memory and finds that M-DRIVE is the current active MOP labeled by EA. In BORIS' semantic memory, a car accident is partially represented as:

```
(M-VEH-ACCIDENT
  TYPE          MOP
  NODE          TEMPLATE
  ROLES         (VEH-OPERATOR ACC-VEHICLE VICTIM)
  MAINCON       EV-COLLIDE
  GOALS         (G-AGENT-P-HEALTH G-P-HEALTH)
  LCONTEXT      PTRANS
```

There is a bottom-up link called DRIVE->VEH-ACCIDENT which connects M-DRIVE to M-VEH-ACCIDENT in the following manner:

```
(DRIVE->VEH-ACCIDENT
  TYPE          MOP-LINK
  NODE          TEMPLATE
  IN            M-DRIVE
  M-LINK-CLASS  BOTTOM-UP
  CONNECTS      M-VEH-ACCIDENT
  ROLE-EQS      [ (DRIVER .  VEH-OPERATOR)
                  (VEHICLE .  ACC-VEHICLE) ]
```

This represents the knowledge that car accidents occur within driving contexts. But there is no such link from the representation of "fighting someone" to M-DRIVE. So the parser decides that the second interpretation is more reasonable than the first.

The word "avoided" can also help to disambiguate "hit" to mean car accident because "avoid" always expects to be followed by an unintentional or undesirable act with respect to the actor. Since a car accident is both unintentional and undesirable under normal circumstances, the second interpretation of "hit" would be chosen by the parser.

3.2.2 Accessing Dynamic Memory Requests

A more interesting type of memory interaction is the parser's ability to look at the active memory requests on the EA agenda list. The parser is aware of what EA expects to see from the story, so context is playing a crucial role in conceptual analysis.

Consider this example:

Unfortunately, the news was not good.

If this sentence were given without the first paragraph, the reader might even have trouble with the first word. "Unfortunately" requires prior context to relate the unfortunate event to. Moreover the reader would probably infer that the news is from a news report on TV or radio. However, given the story the reader automatically knows that "news" refers to the content of the letter. How does the program make this inference?

When the story understander sees letter-arrival, it creates a top-down expectation about the letter's content. As the parser reads about the "news", it notices the EA agenda has a request for a mental object and "news" is certainly one. So the parser signals the entrance of the letter, although the story doesn't explicitly say so. As a result, the follow is representation, produced:

```
(AFFECT
    ACTOR    (NIL)
    OBJ-TYPE (AVERSIVE)
    CON  (MTRANS
             ACTOR (NIL)
             MOBJ (INFORMATION
                      TYPE  (AVERSIVE)
                      LOCAL-CONTEXT (EP1-ROLE3))))
```

Notice that the LOCAL-CONTEXT slot is set to EP1-ROLE3 which is the message within the letter. Local contexts are discussed in 4.3.2.

Agenda-based processing is also useful in inferring role bindings. Consider the following sentence from the story:

Richard eagerly picked up the phone and dialed.

This sentence does not tell us who Richard was calling. However, the reader assumes it to be Paul. When EA processed "Paul gave his home phone number in case Richard felt he could help" in the previous paragraph, it created a positive expectation for Richard to call Paul on its agenda request list. When the parser reads the current sentence, it creates the following representation:

```
(M-PHONE
    CALLER Richard
    CALLEE nil
    EVENT  (EV-MAKE-CALL))
    AFFECT (AFFECT IS    (POS)
                   ACTOR (RICHARD0))
```

At the end of the sentence, the parser still cannot find the slot-filler for CALLEE, so it searches the agenda requests list and finds that a M-PHONE was already expected to occur. Moreover, since the CALLER is Richard as predicted, it makes the inference that the CALLEE is Paul. The search process here is very similar to the search process during Q/A except that, during Q/A, the episodic memory will be searched instead of the agenda request list.

3.2.3 A Comparison: Memory-Based Versus Agenda-Based

The two types of parser/memory interactions discussed above have overlapping functions. They are both helpful in inferring role bindings: the M-BORROW example illustrated a memory-based scheme whereas the M-PHONE example illustrated anaagenda-based scheme. When implicit information is retrieved from episodic memory, we don't have to worry about possible errors in that information. For example, once we are told who the BORROWER and the LENDER in M-BORROW are, we don't expect the role bindings to change unless there is another incidence of M-BORROW. However, inferences based on top-down agenda requests may be erroneous [4]. When EA expects Richard to call Paul, it is still possible that Richard has called somebody else. For example, our story could have been:

> Richard eagerly picked up the phone and dialed. He thought his best friend Bill, who was also a lawyer but specialized in divorce cases, would be in a better position to help Paul. When Bill answered the phone, Richard explained the whole story to him and...

In this case, the CALLEE of M-PHONE should be Bill, not Paul. This type of erroneous inference can be circumvented by tagging all the agenda-based inferences. Whenever such inferences are found to be inconsistent with the explicitly stated values, the explicit information will overwrite the inferred values.

4. EVENT ASSIMILATION

The event assimilator (EA) is the module of the BORIS system responsible for encoding a representation of the story as sentences are read. This representation has been referred to as episodic memory. Encoding is really an inferencing process. As each goal, event, or state comes into the EA, it must fit the added information into the current representation of the story. To do this, inferences must be made to relate the incoming concept to other concepts already in the representation.

Consider for example this passage from the second paragraph of the divorce story:

> Unfortunately, the news was not good. Paul's wife Sarah wanted a divorce. She also wanted the car, the house, the children, and alimony. Paul wanted the divorce, but he didn't want to see Sarah walk off with everything he had. His salary from the state school system was very small.

In processing the final sentence, the EA must infer that having a low salary relates to Paul's desire to protect his possessions. Without making this inference, the EA could not understand the relevance of this sentence to the divorce. It therefore could not relate the low salary information to the divorce in the story representation.

EA is broken down into three processing units: the bottom-up processor, the top-down processor, and the meta-story knowledge monitor. These units work together to build the episodic memory representation of the story, including the episode nodes, the thematic nodes, the scenario nodes, and the semantic links that tie these nodes together.

4.1 Bottom-Up Processing

The bottom-up processor in the BORIS event assimilator is charged with the task of applying context-independent rules. These bottom-up rules may be subdivided into three different classes, based on the the circumstances under which they are applicable and the effect that they have during processing. These include episodic memory creation, goal/plan processing, and scenario mapping.

4.1.1 Episodic Memory Creation

One circumstance that calls for the access and application of bottom-up rules arises when EA cannot directly fit the incoming concept into any current episodes. This always occurs when BORIS processes the first sentence of a new story as in:

Richard hadn't heard from his college roommate Paul for years.

Since this is the first sentence, there do not yet exist any top-down expectations. However, EA is able to build a relationship structure, capturing the roommate relationship between Richard and Paul. When this structure is built, many stereotypical top-down expectations are brought in from the semantic memory knowledge structure for roommates. These expectations are then able to assimilate the concept of "not having heard from" during top-down processing. The next phrase in the story:

Richard had borrowed money from Paul,

is the first event that is described in the story. In this case, EA is able to build a new episodic structure for the borrowing episode. This structure includes expectations relevant to financial agreements which it is able to obtain from the borrowing MOP.

These examples illustrate the most trivial class of episodic memory creation which arises when the parser is able to directly access a semantic memory knowledge structure, such as a MOP via the explicit words used in the passage. The EA then merely creates an episodic instantiation based upon the knowledge structure. Of course, this only takes place when the event cannot be explained via a top-down process.

Another, less trivial example of episodic memory creation is illustrated by the following phrase from the first paragraph of the divorce story:

When a letter arrived from San Francisco,

In this example, a bottom-up rule is associated with the token for "letter" which checks to see if the letter is the object of a PTRANS. If it is, the rule accesses the letter MOP, which eventually leads to the creation of an episode structure for letter sending, friendly communications, etc.

The preceding examples show that bottom-up rules associated with events and tokens can serve as indices to semantic-memory knowledge structures. Memory-accessing rules are also associated with goals, states, and various types of thematic predicates. One should bear in mind however, that the bottom-up process causes only empty templates to be created in episodic memory. It is the top-down process that must fill in these templates.

4.1.2 Goal/Plan Processing

Another circumstance under which bottom-up rules are accessed and applied arises in goal/plan processing. The BORIS event assimilator uses a set of goal/plan rules to monitor incoming concepts for key goal situations such as goal conflict and goal competition [16]. (BORIS also uses top-down mechanisms for goal monitoring which is described in section 4.2.) The following sentence is taken from the second paragraph of the divorce story:

> Paul wanted the divorce, but he didn't want to see Sarah take everything he had.

In the second phrase, we see what may be described as a "non-goal." The following bottom-up goal rule interprets this non-goal in a reasonable way:

```
IF person P1 is said not to have goal G
  which involves an ACT by P2,
THEN there is GOAL COMPETITION
   between P1 and P2 regarding the goal
   to be achieved by the ACT
```

Its application allows the event assimilator to interpret the incoming state as a particular instance of goal competition. The competition is then readily picked up by a divorce expectation during top-down processing.

Another goal/plan rule is illustrated by the following example of a "non-event" from the first paragraph of the divorce story:

> Richard had borrowed money from Paul, which was never paid back, but now he had no idea where to find his old friend.

The second phrase above is an example of how a non-event can be used to generate an expectation for a planbox failure [13]. The following rule reflects how non-events are often related to failed goals:

```
IF an event E is said to not happen,
  AND there is an active goal G which intends E,
THEN form an expectation for a prerequisite
  failure for the event E. If the prerequisite
  failure is found, then mark G as blocked-by
  that failure.
```

The use of this rule allows BORIS to realize that Richard's failure to pay Paul is explained by his inability to find Paul. Furthermore, in the application of this rule, the planbox prerequisite state is indexed by the ATRANS, which specifies the setting enablement required before ATRANS can occur.

4.1.3 Scenario Mapping

The scenario map is a temporally and spatially oriented episodic-memory access structure. It consists of a time-ordered, linked list of scenario descriptors. Each of these descriptors encodes the characters involved in and the events taking place at one particular setting.

The creation of the scenario map entails monitoring the characters as they progress from one location to another throughout the story. A few very simple bottom-up rules can sometimes be useful in this monitoring process. Consider the following hypothetical passage.

> Pete was enjoying his new book when suddenly he remembered e that he had to pick up his children at school. He quickly ran out of the park.

Notice that the context created by the first two sentences in no way indicates where Pete is while he is reading the book. In fact, the default inference might be that he is at his home. The third sentence however conclusively indicates that he was reading in a park. The BORIS parser would represent this last sentence with something like the following:

```
(PTRANS ACTOR  Pete
        OBJECT Pete
        FROM   -park-
        TO     -school-
        ...)
```

The CD predicate PTRANS serves as an index into a set of scene transition rules. One of these rules is sensitive to the FROM slot and will attempt to determine the location of the current scene. This rule is described as follows:

```
IF the OBJECT slot is filled by a
   character in the story,
  AND the FROM slot specifies a location,
  AND this location is not contained
   within the currently known scene location,
THEN this location is the previous
   scene location.
```

There are other analogous rules for determining the location of the next scene in the scenario map. These rules are based on explicitly stated locations given in the story. Consequently, they will override any existing theory of what the current or next scene locations might be.

Other bottom-up, scenario rules are equally simple. These rules monitor such events as new characters entering a scene, characters leaving a scene, and episodes occurring while characters are in transition from one scene to the next. All of the scenario-mapping, bottom-up rules are indexed by the predicate of the concept produced by the parser.

Most of the scenario-mapping process does not, however, occur in the application of bottom-up rules. Scenario mapping is largely driven by the top-down event explanation process that will be described in the next section. The activation of knowledge structures and the use of meta-story knowledge during event assimilation generally provide most of the information needed to generate the scenario map.

4.2 Top-Down Processing

The top-down event assimilation processor in BORIS must access a wide variety of knowledge sources to facilitate the creation of an episodic memory representation of the story. Furthermore it must integrate these various knowledge sources in order to fully understand episodes that fill more than one perspective in the story. Only the knowledge structures that are relevant to a particular episode should be activated. EA must avoid the proliferation of useless inferences that can be associated with the undirected application of any and all associated knowledge structures.

4.2.1 The BORIS Process Model

Top-down event assimilation in BORIS is a process of episodic and semantic memory search. Recall that episodic and semantic memory are intertwined with MOPs. As a concept comes in from the parser, EA must search the active context of the story in order to determine how the concept fits in. This search process is based on a notion of what constitutes the active context of a story as sentences are being read.

The active context of a story is defined in the BORIS system by the following simple recency principle:

> PRINCIPLE OF ACTIVE CONTEXT: The active status of an episode is based on two recency factors:
> 1) when the episode was created,
> 2) when the episode was used to make an inference.

The heart of the EA in BORIS is the episodic memory buffer or the EP-LIST. The EP-LIST is maintained as an ordered list of the episodes that are generated in the story representation during the understanding process. This list is kept ordered on the basis of the principle of active context. As episodes are generated, they are added to the beginning of the EP-LIST. The list is rearranged throughout the understanding process to explain incoming events as some episodes are referred to more frequently than others.

Consider for example, the first two paragraphs of the divorce story:

> Richard hadn't heard from his college roommate Paul for years. Richard had borrowed money from Paul which was never paid back, but now he had no idea where to find

> his old friend. When a letter arrived from San Francisco, Richard was anxious to find out how Paul was.
>
> Unfortunately, the news was not good. Paul's wife Sarah wanted a divorce. She also wanted the car, the house, the children, and alimony. Paul wanted the divorce, but he didn't want to see Sarah take everything he had. His salary from the state school system was very small. Not knowing who to turn to, he was hoping for a favor from the only lawyer he knew.

By the time BORIS has processed the first paragraph, it has generated a borrowing episode and a letter episode. A divorce episode is generated during processing of the second sentence of the second paragraph. At this point the EP-LIST contains:

```
EP-LIST = [ <divorce episode>
            <letter episode>
            <borrowing episode> ]
```

The last sentence of the second paragraph is understood using two of these existing episodes: the divorce episode and the borrowing episode. Contracting a lawyer is understood in terms of the divorce. A return favor is understood as fulfilling an obligation from the borrowing episode which took place long ago. The borrowing episode is brought to the forefront since it is used to process this incoming event:

```
EP-LIST = [ <borrowing episode>
            <divorce episode>
            <letter episode> ]
```

One can view episodic memory in BORIS as a network of interconnected episode and thematic nodes as shown in Figure 4-1.

As a concept comes in from the parser, EA searches through this network of nodes for an episode that will explain the new concept. The search order of these nodes is determined by their

```
        +-------------------------------------+
        |                                     |
+-------|--+       +----------+               |
| EP0      |       | EP1      |               |
|Borrowing |--+    |Letter    |-----------+   |
+----------+  |    +----|-----+         +--|---|---+
              |         |               | EP2      |
              |         |               |Divorce   |
              |    +----|-----+         +---|--|---+
              |    | REL0     |             |  |
              +----|Friends   |-------------+  |
                   +----------+                |
                                        +----------+
                                        | REL1     |
                                        |Marriage  |
                                        +----------+
```

Figure 4-1: EPISODIC MEMORY NETWORK

ordering on the EP-LIST. Consequently, more active episodes will be searched first. The search process continues until the incoming concept has been explained. The overall algorithm of EA is shown in Figure 4-2.

4.2.2 Concept Explanation

An episode consists of a set of partial instantiations of interrelated knowledge structures. Each of these knowledge structures brings its own interpretation of the events within the episode along with its own set of expectations pertaining to the episode. Consider, for example, the restaurant episode in the divorce story.

There are expectations associated with the restaurant MOP. A waitress will probably come to serve the patrons. They will probably order from a menu, and so on. There are expectations associated with the lawyer MOP (for the legal consultation). Richard and Paul will probably discuss the divorce case. There

```
                  / ____________________ \
                 |   EVENT ASSIMILATOR    |
                  \ ____________________ /
                             |
      +------------->|
      |
      |          +------------+
      |          | More       |-----------------+
      |          | Episodes?  | no              |
      |          +------------+                 |
      |                 |yes                    |
      |      +-------------------+   +----------------+
      |      | Consider the next |   | Apply bottom-up|
      |      | episode in the    |   | explanation    |
      |      | EP-LIST to explain|   | rules.         |
      |      | the new concept.  |   |                |
      |      +-------------------+   +----------------+
      |                 |                     |
      |   no     +------------+               |
      +----------| Is concept |               |
                 | explained? |               |
                 +------------+               |
                        | yes                 |
             +--------------------+           |
             | Further activate   |           |
             | the explaining     |           |
             | episode by moving  |           |
             | it up in EP-LIST.  |           |
             +--------------------+           |
                        |                     |
                        V                     |
                      EXIT <------------------+
```

Figure 4-2: EVENT ASSIMILATOR: The top-down algorithm for event assimilation.

are expectations associated with the meal MOP. They perhaps are eating because they are hungry. There are expectations associated with their friendship and the socializing META-MOP. They are probably excited to see each other again.

Each episode node in the story representation has an agenda associated with it. An agenda is an ordered list of agenda levels, each consisting of a list of agenda requests. An agenda request is a test-action pair which embodies an expectation [12]. The agenda control structure will repeatedly try all of the requests at a given level until none of them "fire" (i.e. none of their tests are true). Then the agenda processor will move on to the next level, repeating the process. An episode's agenda has been fully considered when there are no more levels left to examine.

Expectations are added to an episode's agenda whenever a new knowledge structure is activated by the episode. A new episode is always created by the excitation of at least one knowledge structure. Its initial expectations come from this initiating knowledge structure.

Most expectations associated with the various knowledge structures fall into five categories: 1) expectations which fill slots, 2) expectations which are activated when their associated knowledge structures are referenced, 3) goal/planning expectations, 4) scenario-related expectations, and 5) event expectations.

The most straightforward agenda requests are for slot-fillers. For instance, the divorce MOP expects to find a husband and a wife. The lawyer MOP expects to find a lawyer and a client. These requests are activated by explicitly stated role bindings (such as "Paul's wife Sarah").

MOPs and META-MOPs are interconnected in semantic memory via MOP-links. Associated with each MOP-link is an activation

```
               /  ___________________  \
              |   EVENT ASSIMILATOR     |
               \ ___________________   /
                         |
   +--------------->|
   |             +--------------+
   |             |  More         |-----------------+
   |             | Episodes?     | no               |
   |             +--------------+                   |
   |                   |yes                         |
   |          +---------------------+     +-----------------+
   |          | Consider the next   |     | Apply bottom-up |
   |          | episode in the      |     | explanation     |
   |          | EP-LIST to explain  |     | rules.          |
   |          | the new concept.    |     |                 |
   |          +---------------------+     +-----------------+
   |                   |                          |
   |     no      +--------------+                 |
   +---------| Is concept   |                     |
             | explained?   |                     |
             +--------------+                     |
                   | yes                          |
            +---------------------+               |
            | Further activate    |               |
            | the explaining      |               |
            | episode by moving   |               |
            | it up in EP-LIST.   |               |
            +---------------------+               |
                   |                              |
                   V                              |
                  EXIT <--------------------------+
```

Figure 4-2: EVENT ASSIMILATOR: The top-down algorithm for event assimilation.

are expectations associated with the meal MOP. They perhaps are eating because they are hungry. There are expectations associated with their friendship and the socializing META-MOP. They are probably excited to see each other again.

Each episode node in the story representation has an agenda associated with it. An agenda is an ordered list of agenda levels, each consisting of a list of agenda requests. An agenda condition. This condition dictates the circumstances under which connected MOPs are relevant to the episode at hand. For example, the activation condition for bringing MM-LEGAL-DISPUTE into a divorce episode is the existence of goal conflict between the husband and the wife. When M-DIVORCE was used to create the divorce episode, an expectation was created to check for this goal conflict. Later in the second paragraph, the story describes the goal conflict, so the expectation fires and MM-LEGAL-DISPUTE is brought into the episode.

Goal/planning expectations monitor the existence and status of goals involved in a knowledge structure. For example, the divorce MOP has expectations about the desire of the participants to retain the family possessions, since this is usually a very important aspect of divorces.

Some MOPs are partitioned into scenes. For instance, M-LETTER is broken down into a sending scene and an arrival scene. This information is used in the creation of the scenario map to find the transition from one scene to the next. Also, there are default locations associated with certain MOPs. A letter arrival, for example, usually takes place either at the home or the business of the person receiving the letter.

Event expectations look for the occurrence of an event that temporally follows the last instantiated event in the MOP. This process is driven by a pattern matcher and is very similar to script application [1].

The explanation algorithm simply uses the agenda control structure to run the agenda associated with an episode. Expectations are tied to an episode via its agenda. Thus, the process knowledge in BORIS is distributed throughout Episodic memory. As new episodes are created, new agendas are also created and associated with them. Thus, agendas in BORIS grow dynamically. As episodic memory is searched, the relevant process knowledge is applied. The top-down mechanism of EA involves trying the agenda requests associated with the episodes being searched. Therefore, the process of understanding and the process of memory search are closely related.

4.3 Meta-Story Knowledge Monitor

BORIS is able to take advantage of knowledge pertaining to how people organize stories. This currently manifests itself in two ways: via main character identification and via the establishment of local contexts.

4.3.1 Main Character Identification

A paragraph in a story often reflects the perspective of one of its characters, revolving around his goals and actions. BORIS attempts to identify the main character in each paragraph to drive expectations based on his relationships with the other characters in the story.

The main character in a paragraph is usually identified by the parser, according to several heuristics: For instance, the main character might be specified in the dominating concept generated to represent the first sentence of a paragraph. Or the perspective of relationships described in the story could reveal the identify of the main character. The phrase "Paul's wife Sarah" is evidence that Paul is the main character in the second paragraph.

The main character often initiates episodes in a paragraph. For example, because Richard initiates the driving episode and the meeting/restaurant episode in the fourth paragrpah, BORIS understands Richard as being the main character in that paragraph.

The identity of the main character is used to organize interpersonal thematic expectations. An agenda is associated with each of the relationships and interpersonal themes between the different characters in the story. We might, for example, expect two friends to get together periodically.

When the main character of a paragraph is identified, his relationships with the other characters in the story are brought to the forefront. Subsequently, the agendas for these relationships are considered during EA until the remainder of the paragraph has been read. Thus events are understood in terms of the thematic concerns of the main character in each paragraph.

4.3.2 Local Contexts

In the course of a story, the writer usually provides the reader with commentary or setting information. He may reveal the thoughts of one of the characters or give background information pertinent to the relationship between two of the characters.

In the second paragraph of the divorce story, the writer describes the contents of the letter written from Paul to his friend Richard, which is another embedded story. The understander must realize that everything described in the second paragraph must be grouped together as the mental object of the Letter Episode initiated in the first paragraph. There would be an analogous problem in a phone conversation, an intercom message, etc..

BORIS is able to deal with this situation by entering a local-context mode. In this mode, all the episodes that are created are bunched together as a large mental object unit. The system can then refer to this unit as a whole. In the letter episode this unit fills the object-slot of the MTRANS concept. BORIS enters local-context mode when the first sentence of the following paragraph refers to a mental object unit. So in this example, the potential for a local-context mode is created when the Letter MOP is activated in the first paragraph.

The importance of recognizing local contexts appears when BORIS infers that Richard is a lawyer. If we examine the story carefully, we can see that this is never explicitly stated. Rather, it reads:

He was hoping for a favor from the only lawyer he knew.

The rule BORIS uses to make this inference is the following:

```
IF person P1 MTRANSes to person P2
  that P1 has a goal requiring an AGENT
  AND the AGENT is never mentioned,
THEN P2 is the likely AGENT.
```

In order to apply this rule, BORIS must be aware that the request for a favor is being implicitly MTRANSed to Richard, since Richard is in the local context of reading Paul's letter. It is the local context which supplies this awareness.

5. QUESTION ANSWERING

Many heuristics useful to Q/A have been described elsewhere [8]. Given this background, we will cover only those aspects of Q/A which are unique to the BORIS system.

5.1 Understanding at Q/A Time Versus EA Time

BORIS uses the same parser to process both the sentences in the story and any questions about the story. Since the parsing is integrated with the memory, in most cases Q/A processing proceeds in a similar manner. However, there are three situations in which the "mode" (Q/A vs EA) of understanding effects the parser's actions.

5.1.1 *Tokenization*

Whenever the parser encounters a reference to an object primitive, a setting or a character, memory is searched to find out if the object already exists in episodic memory. In EA mode, the failure to find a referent results in the creation of a new token in memory. However, during Q/A, a failure to find a referent results in abandoning any attempt to find an answer. Instead, a 'complaint' is generated to point out that a referent could not be found. For example,

> Q: What does George do for a living?
> A: I don't recall any mention of a character *named George in the story.*

5.1.2 Presupposition Checking

During Q/A the parser checks the role bindings presumed by the question against the actual bindings maintained in episodic memory. During EA, if the bindings fail to match, it indicates that a new instantiation must be built. For example, the mention of a letter from Sarah would cause BORIS to create a new M-LETTER instance, separate from Paul's letter. During Q/A, however, a failure of roles to match indicates a false presupposition so BORIS rejects the question and corrects the false presupposition. For instance:

> Q: Why did Sarah write to Richard?
> A: It was Paul, not Sarah, who wrote to Richard.

5.1.3 Question Words.

The parser handles words like "who", "what", "why", "where", etc. in special ways. If such a word is encountered during Q/A, it is assumed to be initiating a request for information. During EA however, special demons process these words to indicate the beginnings of clauses. If a question is encountered while the story is being processed, then it is treated as a rhetorical question, and no attempt is made to answer it.

5.2 Using Memory To Understand

For BORIS, understanding a question means finding a referent to the question concept in episodic memory. One consequence of this view is that episodic memory is examined not only while the question is being answered, but also while it's being understood. Unlike many data base retrieval systems, which have an isolated parser as a front-end, BORIS examines story-memory to aid in the understanding process itself. For example, when people are asked:

> Q: Why didn't Richard pay Paul back?

they never even consider that "pay back" might refer to anything other than the return of loaned money because they are thinking of the story while they are parsing that question. If, however, the story had read:

> Richard had an affair with Paul's wife and Paul found out. Paul vowed that he would kill Richard for having ruined his marriage. Paul went after Richard with a shotgun, but the police put Paul in jail because they thought he was having a nervous breakdown. As the police carried him away, Paul vowed that he would punish Richard even if it was the last thing he did.

Then when people are asked:

Q: Why didn't Richard pay Paul back?

they won't even consider that "pay back" might mean return of money. They immediately understand it in terms of revenge.

"Pay back" can not be disambiguated by syntactic or semantic (i.e. general world) knowledge. What, then, is a reasonable representation for "pay back"? Obviously, one must know what happened in the story. Thus, the result of parsing "pay back" in this episodic context leads to a representation which refers to an event in episodic memory. By the time the question is understood, then, episodic memory has already been searched. This leads to more rapid recall of an appropriate answer. As a result, sometimes the answer to a question will be known by BORIS before the question has been completely parsed.

Episodic search is especially useful when the questions are "vague". For example, "Did Paul contact Richard from San Francisco?" is less specific than "Did Paul write to Richard from San Francisco?" In such cases, episodes involving META-MOPs are searched. When asked:

Q: Had Paul helped Richard?

BORIS searches for a META-MOP in episodic memory which involves agency. Once MM-FAVOR has been found as a likely referent, the specific event associated with this abstract event can be accessed:

A: Yes. Paul had lent Richard money.

5.3 Reconstructive Search

BORIS instantiates only fragments of MOPs. This is because search processes at retrieval-time are able to reconstruct events from a given MOP. When BORIS is asked:

Q: Did Paul write to Richard?

it searches M-LETTER in episodic memory. Since the writing of a letter was never mentioned, there is no instantiation for such an event in episodic memory. However, this event can be inferred. BORIS knows that Richard could not have received a letter from Paul unless Paul had written a letter. At this point in the search, the event of Paul's writing a letter is retrieved and instantiated. The next time this question is asked, the instantiation will be found (not inferred). This means that asking the same question again will result in answering it more quickly the second time. Thus BORIS's episodic memory may be altered during the question-answering process. This allows for an increase in the efficiency of recall, since BORIS only instantiates what is used for retrieval.

6. GENERATION OF ENGLISH IN BORIS

The BORIS generator is based upon the system GEN, written by Rod McGuire at Yale [11]. A few changes to GEN were necessary to make it work well in the BORIS system. For instance, GEN assumed that its input would be a precise conceptual representation. In BORIS this is rarely true -- the memory structures contain much more information than it is desirable or feasible to express. In fact all of BORIS memory is linked, and "complete" generation of one concept would require that all of memory be expressed. Furthermore, the BORIS memory structures have been developed specifically to facilitate the internal processes of understanding, with generation being a secondary consideration. Therefore these structures are in many ways unsuited to direct generation.

6.1 The Basics of Generation

Generation is accomplished by processing elements of a "phrase-stream" [11], or list of sentence units to be expressed. The generator loops, dealing with these elements until the phrase-stream is empty. Phrase-stream elements contain generator instructions and provide access to other instructions (stored as data) which eventually access words. How these elements are derived and used is explained later, but it is important to recognize that they must access memory. One of the most significant features of the BORIS generator is the way this lexical memory is organized. The organization is based upon a hierarchy of lexical prototypes developed in McGuire's GEN, and adapted and expanded for use in BORIS.

6.1.1 The Prototype Hierarchy

The prototype hierarchy is a system for classifying generation information according to a hierarchy of lexical "prototypes" which the generator uses for information on how to express a concept. The hierarchy is tree-like in that the higher-level elements may have each many subordinate elements. Raw concepts are treated as the terminal nodes of the tree.

In BORIS, generation information is not stored in individual conceptual instantiations (as in GEN). Rather, it is associated with instantiation "templates" in the prototype hierarchy. Since the generator needs lexical information to be able to express a concept, it has to access this information by examining the hierarchy. Templates frequently have part of the information necessary for generation, but usually the generator must look for at least some of the information higher up in the tree.

The prototype hierarchy is intended to consolidate information about similar expressive forms into a single memory unit. When a representation (such as CD) has a characteristic that can be treated as equivalent for generation, the database used by the generator should only store that information in one place. Thus, the more common information is stored in the trunk of the tree and the more specific in the branches.

Most concepts have a lexical prototype and the process of tracing up the various levels of the hierarchy ultimately leads to the lexical entity G-S (for "goal/state" in GEN). G-S is an "orphan", in that it has no prototype (parent) node.

Stored in this set of prototypes is information about how concepts should be expressed. This might be as simple as a particular word to use, or verb information referring to a table of forms for that particular verb. More complex data is also used, such as the order in which to express concepts, paths to follow to check whether a concept should be expressed, and alternatives to use when some paths may fail.

The generator searches the prototype hierarchy in the following manner: In order to find an item of generation information, it checks to see whether the raw concept (the input concept) contains this information. If the information is there, no more searching need be done. Otherwise, the lexical prototype is looked up. If there is no lexical prototype then it is assumed that the current item is to be expressed as is. If there is a lexical prototype but it doesn't contain the proper information, the next prototype is checked (the prototype of the prototype), and so forth.

6.1.2 The Process of Generation

The fundamental dynamic element of the generator is the "phrase-stream" [11], which organizes the expression of the various parts of the current concept and keeps track of exactly when to perform various actions. These actions are pointers to LISP code which ultimately produce "lexical items" or words.

The generator proceeds by expanding the expression on the left of the phrase stream and replacing it by its expansion. This expansion usually involves a search of the prototype hierarchy. When a phrase-stream element cannot be expanded, it is considered to be a word, removed from the phrase-stream and placed in temporary storage. The cycle is then repeated on the new phrase stream. Generation ends when all expansions are completed and the phrase stream is empty. The remaining task of the generator is to print the words.

The structure of the phrase-stream results in a mode of generation similar to left-to-right, depth-first traversal of a syntactic tree. However syntactic constructs are determined solely on the basis of conceptual content and the choice of words and phrases is controlled by the conversational context as well as by a conceptual "dictionary."

It is the prototype hierarchy which controls the actual sequence of subject, verb, phrasal-object and other parts of a sentence. At first the phrase-stream contains only a single element, placed there at the beginning of a generator run, which expands to the correct sequence of expressions according to the concept being generated. The original element of the phrase-stream functions to search the prototype tree for the contextually, conceptually and lexically appropriate method of expression. This element is expanded into several elements which organize the expression. At each stage of expansion, contextual and semantic factors can be taken into account.

The prototype hierarchy can be seen as a discrimination net for the expression of different concepts. All expression is controlled by the lexical-prototype system with the phrase-stream being completely subordinate to this data-base. The initial element of the phrase-stream is a pointer to a section of LISP code which finds a phrase structure for the concept being generated. This is done by finding the most specific lexical prototype which contains information about phrase structure (i.e. searching the prototype tree until such a prototype is found.)

Many concepts require a similar phrase structure. For these concepts the proper information is stored in a common lexical prototype. For cases where special requirements are made of the generator, the unique code is stored low in the prototype tree.

An example of the use of the phrase-stream and the prototype hierarchy to generate English from a BORIS "PTRANS" concept appears in figure 6-1. In this figure, EV-POSTAL refers to the event of a letter being brought by a mailman. This is represented in Conceptual Dependency as:

```
(PTRANS ACTOR mailman
        OBJECT letter
        FROM mailman
        TO  reader)
```

EV-POSTAL0 is a BORIS memory node constructed to represent the event of the arrival of Paul's letter in paragraph one of the divorce story. When the BORIS generator expresses EV-POSTAL0, it finds no generation information in the instantiated node, so it traverses the prototype link to the template, EV-POSTAL, a node in the letter MOP. The template node contains two items of information: 1) use the verb "to get", and 2) subject of the sentence will be found in the TO slot of the conceptual

```
                                         prototype nodes:

top level node:                                  G-S
      .                                        /  |   \
      .                                     DO   STATE  AFFECT . .
      .                                   / | \                .
      .                                  /  |  \                .
PTRANS lexical node:           PTRANS   ...   INGEST             .
                                /|\                  .
                                 |                    .
BORIS PTRANS template:       EV-POSTAL                 .
                               /|\
                                |
instantiation:              EV-POSTALO
```

Figure 6-1: Sample Hierarchical Structure and Traversal To Seek Generation Information

representation. The lexical prototype of EV-POSTAL is PTRANS, which has certain PTRANS-specific information about phrasal-object expression. PTRANS then has a prototype link ultimately to G-S, which contains some information which is overridden by information in more specific nodes. G-S also organizes the sentence structure in a fairly standard way. Using the role bindings in the instantiation, the concept is ultimately expressed as: "RICHARD GOT A LETTER FROM PAUL."

6.1.3 The Role of a Generator

The role of a generator is to be a means to expression. All aspects of expression, from finding the words for a concept, to organizing several concepts coherently, should be the responsibility of generator-type functions. GEN takes care of basic expression: In BORIS, the task of expression is more complex, and is partly dealt with in the way the Q/A mechanism organizes conceptual answers to questions.

6.2 Generation Research In Progress

Generation is often viewed as the last step after understanding and response formation. It is important to recognize, however, that every text we parse has been generated. This view of generation helps emphasize the true role of generation; we generate in order to be understood. Below are some of the directions being taken :

1. INTEGRATED GENERATION: People answering questions often begin to express their answers before they have n been fully determined. In BORIS, what remains difficult is the generation of appropriate transition expressions to make the fragmentary output appear "smooth". This argues for a merging of generation tasks with understanding understanding tasks.

2. SUMMARIZATION: In a broad-based natural language system, we find tasks which involve similar processing at various levels. In programming terms, story-parsing and question-parsing in BORIS are done by the same function, operating in slightly different modes. It is reasonable to assume that people have one language understanding mechanism which they apply to all their language understanding tasks. Likewise, it is reasonable to assume that expressive tasks will be based upon a single generator function. In BORIS there are two generation tasks. The prime application is in Q/A but it has also been used in summarizing and paraphrasing. The summarization process is based upon Wendy Lehnert's plot unit system [9] which organizes memory according to certain states (positive acts, negative acts and mental states). Plot units are specific combinations of these states which are distinct and important enough that they can be treated as individual units. Once a program has built memory based upon plot units, they can be analyzed to produce a summary of the story.

3. GENERATION FROM TAUS: Currently, generation makes use of MOPs and CD primitives. TAUs are used in generating responses to questions involving affects. TAUs also have expression information directly associated with them. Future work will allow BORIS to generate adages from TAUs. For example, in answer to the question: "What is a moral of the story?" a response from TAU-HIDDEN-BLESSING would be: "Every cloud has a silver lining."

4. DIALOG PRIMITIVES: Research is also underway to expand the existing set of "dialog primitives" used in BORIS. Dialog primitives capture the speakers overall intentions in generating his response (in addition to its specific content). For example, the predicate DISAGREE represents the speaker's motivations for expressing disagreement, as in arguments of fact or interpretation. Such primitives have generational frames associated with them. For instance, the following frame:

   "BUT IT WAS (person) WHO (role),
    NOT (other person) ..."

   is associated with the dialog primitive, DISAGREE.

7. FUTURE RESEARCH AND CONCLUSIONS

7.1 Future Work

There are numerous directions for BORIS to go in. Much work remains to be done in representing the kinds of knowledge needed to understand stories like the divorce story, and to increase the interaction between knowledge sources. For example, the distinction between the event assimilator and the parser is somewhat artificial. Therefore, a new version of BORIS is being designed which does away with this distinction -- leaving a system that is completely integrated.

Other research directions include: improving the program's use of memory overlays and multiple perspectives, expanding the roles of TAUs, and exploring the effects of episodic memory upon processing.

The next immediate step in our research methodology involves testing the BORIS system on variations of the divorce story, in order to improve the robustness of the system and to see what kinds of modifications are necessary to both the representational structures and process mechanisms.

One good sign has been that each successive paragraph in the divorce story has taken less time and effort to implement.

7.2 Conclusions

It is difficult to appreciate the complexity involved in natural language processing because people do it so effortlessly. Anyone can read a newspaper or engage in a conversation. Yet the goal of producing commercial natural language systems remains unattained.

People are the most impressive examples of intelligent processors which employ many divergent sources of knowledge operating at varying levels of detail and interaction. Currently, the only areas which directly exercise and reveal such uniquely human skills are to be found in natural-language-based tasks: i.e. conversation, question answering, translation, argumentation, and story understanding. As such, BORIS's most important contribution is as both a source and testbed for theoretically interesting problems concerning such basic cognitive skills.

8. APPENDIX

BORIS runs on a DEC-20 at Yale and is implemented in TLISP (Yale UCI Rutgers LISP). What follows are fragments of BORIS execution traces. These traces have been superficially edited in order to shorten their length and improve readability. The Q/A trace shows the parsing process in the most detail.

8.1 Trace of BORIS on First Paragraph

```
ACTIVE MODULE: CONCEPTUAL-ANALYSIS
(PGH) RICHARD
CON0 = HUMAN NAME (RICHARD) GENDER (MALE))

HADN'T  HEARD FROM
CON5 = (MTRANS ACTOR (NIL) TO RICHARD0 TIME (PAST) MODE (NEG))

HIS COLLEGE ROOMMATE PAUL FOR YEARS (PRD)
CON13 = (HUMAN REL (R-ROOMMATES ROOMMATE-A PAUL0
               ROOMMATE-B RICHARD0 ERA (COLLEGE))
               GENDER (MALE) NAME (PAUL))


ACTIVE MODULE: IP-RELATION-CREATION
== NEW INTERPERSONAL RELATION ==   R-ROOMMATES0
== INFERRING THEME FROM RELATION ==
== NEW INTERPERSONAL THEME ==   IPT-FRIENDSHIP0

ACTIVE MODULE: CONCEPTUAL-ANALYSIS
RICHARD
CON24 = (HUMAN NAME (RICHARD) GENDER (MALE))

HAD BORROWED
CON27 = (M-BORROW MB-LENDER (NIL) MB-BOBJECT (NIL)
                 MB-BORROWER RICHARD0 EVENT (EV-LEND-OBJ))

MONEY
CON32 = (MONEY)

ACTIVE MODULE: EPISODE-CREATION
== NEW EPISODE ==    EP0
== CREATING PERSPECTIVE ON EP0==    M-BORROW
ACTIVE MODULE: EVENT-ASSIMILATION
== CONSIDERING EPISODE EP0 ==
== EXPLANATION ==  Event explained in EP0
   Perspective:  EV-LEND-OBJ in M-BORROW

ACTIVE MODULE: CONCEPTUAL-ANALYSIS
```

```
FROM PAUL
CON27 = (M-BORROW MB-LENDER PAULO MB-BOBJECT TOK1
                  MB-BORROWER RICHARDO EVENT (EV-LEND-OBJ))

WHICH WAS NEVER PAID BACK (CMA)
CON40 =  (ATRANS ACTOR RICHARDO OBJECT TOK1 TO PAULO MODE (NEG))

ACTIVE MODULE: EVENT-ASSIMILATION
== CONSIDERING EPISODE EP0 ==
== CREATING PERSPECTIVE ON EP0==    MM-FAVOR
== EXPLANATION ==  Event explained in EP0
     Perspective:  EV-RETURN-OBJ in M-BORROW

ACTIVE MODULE: CONCEPTUAL-ANALYSIS
BUT NOW HE
CON45 = (HUMAN GENDER (MALE) CONTYPE (PRON)
               CASE-FRAME (NOMINATIVE))

HAD NO IDEA
CON49 = (NEG)

CON50 = (PLAN-BOX-SUCCESS)

WHERE TO FIND HIS OLD FRIEND (PRD)
CON52 = (D-KNOW ACTOR RICHARDO OBJECT TOK2
                STATUS (PLAN-BOX-SUCCESS MODE (NEG)
                              ACTOR RICHARDO TIME (NOW)))

WHEN
CON64 = (TIME-OF CON (NIL))

A LETTER
CON67 = (PHYSOBJ TYPE (LETTER))

ACTIVE MODULE: TOKEN-MOP-SETTING
== NEW EPISODE ==    EP1
== CREATING PERSPECTIVE ON EP1==    M-LETTER

ACTIVE MODULE: CONCEPTUAL-ANALYSIS
FINALLY ARRIVED FROM SAN FRANCISCO (CMA)
CON64 = (TIME-OF CON (PTRANS OBJECT LETTERO FROM SAN-FRANCISCOO))

ACTIVE MODULE: EVENT-ASSIMILATION
== CONSIDERING EPISODE EP1 ==
== EXPLANATION ==  Event explained in EP1
     Perspective:  EV-POSTAL in M-LETTER
== CREATING PERSPECTIVE ON EP1==    MM-COM

ACTIVE MODULE: CONCEPTUAL-ANALYSIS
RICHARD
CON73 = (HUMAN NAME (RICHARD) GENDER (MALE))

WAS ANXIOUS TO FIND OUT HOW PAUL WAS
CON76 = (GOAL ACTOR RICHARDO DESIRE (STRONG)
              GOBJ (D-KNOW
                        ACTOR RICHARDO OBJECT (STATE ACTOR PAULO))
                        TIME (TIME-OF CON (PTRANS
                                           OBJECT LETTERO
                                           FROM SAN-FRANCISCOO)))

ACTIVE MODULE: EVENT-ASSIMILATION
== CONSIDERING EPISODE EP1 ==
== EXPLANATION ==  Goal explained in EP1
     Perspective:  G-KEEP-IN-TOUCH in MM-COM

QA-SESSION:
```

8.2 Trace of Q/A

```
Had Paul helped Richard?

Processing word: HAD
Using word: DID
    Adding to *wm*: CON297
    Spawning: DEM108 = (CONNECT-VERIFY CON297 (ACT GOAL MOP) AFT)
        CON297 = (MODE IS IS10)

Processing word: PAUL
    Adding to *wm*: CON298
    Spawning: DEM109 = (HUMAN-BIND CON298 INSTAN21
                                  PAUL FIRST-NAME)
        CON298 = (HUMAN NAME NAME10
                        GENDER GENDER10
                        INSTAN INSTAN21)
    Executing: DEM109
        INSTAN21 <-- (PAULO)
    Killing: DEM109

Processing word: HELPED
    Using root: HELP with suffix: ED
    Adding to *wm*: CON299
    Spawning: DEM110 = (FRIENDS-MAY-FAVOR CON299)
    Spawning: DEM111 = (EXP CON299 AGENTO HUMAN BEF)
    Spawning: DEM112 = (EXP CON299 ACTOR7 HUMAN AFT)
        CON299 = (AGENCY AGENT AGENTO
                         ACTOR ACTOR7
                         GOAL GOALO)
    Executing: DEM111
        AGENTO <-- CON298
    Killing: DEM111
    Executing: DEM108
        CON299 <-- (AGENCY AGENT AGENTO ACTOR ACTOR7 GOAL GOALO
                           MODE (VERIFY))
    Spawning: DEM113 = (ANS-VERIFY-QUES CON299)
    Killing: DEM108
```

```
Processing word: RICHARD
    Adding to *wm*: CON300
    Spawning: DEM114 = (HUMAN-BIND CON300 INSTAN22
                                   RICHARD FIRST-NAME)
        CON300 = (HUMAN NAME NAME11
                        GENDER GENDER11
                        INSTAN INSTAN22)
    Executing: DEM114
        INSTAN22 <-- (RICHARD0)
    Killing: DEM114
    Executing: DEM112
        ACTOR7 <-- CON300
    Killing: DEM112
    Executing: DEM110

SEARCH INTERPERSONAL THEME BETWEEN RICHARD0 AND PAUL0
     :::>> (IPT-FRIENDSHIP0) FOUND

SEARCH EPISODIC MEMORY FOR INSTANTIATION OF MM-FAVOR
     :::> (EP0) FOUND

REINTERPRET AGENCY AS MM-FAVOR

    Spawning: DEM115 = (FIND-EP CON299 INSTAN230)
        CON299 = (MM-FAVOR FAV-RECEIVER ACTOR7
                           FAV-GIVER AGENT0
                           MODE MODE0
                           INSTAN INSTAN230)
    Killing: DEM110
    Executing: DEM115

SEARCH FOR AN INSTANTIATION OF MM-FAVOR
    :::>> (EP0) FOUND

        INSTAN230 <-- (EP0)
    Spawning: DEM116 = (PRES-CHECK CON299 AGENT0 FAV-GIVER EP0)
    Spawning: DEM117 = (PRES-CHECK CON299 ACTOR7
                                   FAV-RECEIVER EP0)
    Killing: DEM115
    Executing: DEM117
    Killing: DEM117
    Executing: DEM116
    Killing: DEM116

Processing word: *QMARK*
    Adding to *wm*: CON301
        CON301 = (*END*)
    Executing: DEM113

SEARCH MAIN EVENT OF META-MOP MM-FAVOR AS ELABORATION
     :::>> EV-LEND-OBJ0 FOUND
    Killing: DEM113
```

Result of parse:

```
    (MM-FAVOR FAV-RECEIVER (HUMAN NAME (RICHARD)
                                  GENDER (MALE)
                                  INSTAN (RICHARDO))
              FAV-GIVER (HUMAN NAME (PAUL)
                               GENDER (MALE)
                               INSTAN (PAULO))
              MODE (VERIFY)
              INSTAN (EPO))

Answer is (VERIFICATION IS (POS)
                        SEARCH (EPISODIC)
                        ELAB (EV-LEND-OBJO))
```

8.3 Trace of Generation

This trace generates English output for the question processed in section 8.2:

```
             !D following (PHL)
             !S of: (D-PERHAPS) following path: (IS)
YES ...
CMA ...       !S of: (D-PERHAPS (D-PH)) following path: (ELAB)
               !D following (PHL)
               !D following (SUBJECT)
               !S of: (D-SUBJ) following path: (*SUBJ)
                 !D following (FIRST-NAME)
PAUL ...        !D following (MODE LEX)
               !D following (LEX VERB)
LENT ...        !S of: (D-PH) following path: (OBJECT)
                 !D following (LEX)
MONEY ...
TO ...          !S of: (D-PH) following path: (TO)
                 !D following (FIRST-NAME)
RICHARD ...
PRD ...

 YES, PAUL LENT MONEY TO RICHARD.
```

REFERENCES

[1] Cullingford, R. E.
Script Application: Computer Understanding of Newspaper Stories.
Technical Report 116, Yale University. Department of Computer Science, 1978.

[2] DeJong II, Gerald F.
Skimming Stories in Real Time: An Experiment in Integrated Understanding.
Technical Report 158, Yale University. Department of Computer Science, 1979.

[3] Dyer, Michael G. and Lehnert, Wendy G.
Organization and Search Processes for Narratives.
Technical Report 175, Yale University. Department of Computer Science, 1980.

[4] Richard H. Granger, Jr.
Adaptive Understanding: Correcting Erroneous Inferences.
Technical Report 171, Yale University. Department of Computer Science, 1980.

[5] Kolodner, Janet L.
Retrieval and Organizational Strategies in Conceptual Memory: A Computer Model.
Technical Report 187, Yale University. Department of Computer Science, 1980.

[6] Michael Lebowitz.
Generalization and Memory in an Integrated Understanding System.
Technical Report 186, Yale University. Department of Computer Science , 1980.

[7] Lehnert, Wendy G. and Burstein, Mark H.
The Role of Object Primitives in Natural Language Processing.
Technical Report 162, Yale University. Department of Computer Science, 1979.

[8] Lehnert, Wendy G.
The Process of Question Answering.
Lawrence Erlbaum, Hillsdale, New Jersy, 1978.

[9] Lehnert, W. G.
Affect Units and Narrative Summarization.
Technical Report 179, Yale University. Department of Computer Science , 1980.

[10] Schank, Roger C.
Reminding and Memory Organization: An Introduction to MOPs.
Technical Report 170, Yale University. Department of Computer Science, 1979.

[11] McGuire, R.
Political Primaries and Words of Pain.
Unpublished Manuscript, Dept. of Computer Science, Yale Univeristy, New Haven, CT.
1980.

[12] Christopher K. Riesbeck and Roger C. Schank.
Comprehension by Computer: Expectation-Based Analysis of Sentences in Context.
Technical Report 78, Yale University. Department of Computer Science , 1976.

[13] Schank, Roger and Abelson, Robert.
Scripts, Plans, Goals, and Understanding.
Lawrence Earlbaum Associates, Hillsdale, New Jersey, 1977.
The Artificial Intelligence Series.

[14] Schank, R. C. and Jaime G. Carbonell, Jr.
Re: The Gettysburg Address. Representing Social and Political Acts.
Technical Report 127, Yale University. Department of Computer Science, 1978.

[15] Schank, Roger C.
Conceptual Information Processing.
North Holland / American Elsevier, 1975.
Fundamental Studies in Computer Science, Volume 3.

[16] Wilensky, Robert.
Understanding Goal-Based Stories.
Technical Report 140, Yale University. Department of Computer Science , 1978.

DATE
ILME